Computer Programming 2
(Introduction to OOP & Classes)

Dr. Dina M. Ibrahim
1st year Students
2014-2015

Lecture #5

14/3/2015

Topics

- Basic definitions of OOP
- Evolution of Programming Language and History of C++
- Fundamental concepts on OOP
- Classes and Instances
- What is UML?
- Defining methods of a Class

Basic Definitions of OOP

- ***Definition 1:*** An **Object** is any thing, real or abstract, about which we store data and those operations that manipulate the data. *An object is a representation of information* consisting of private memory, and a set of operations to manipulate information stored in the private memory or to carry out some action relative to that information.

- ***Definition 2:*** **Data Encapsulation** or **information Hiding** is the concealing of the implementation details of a data object from the outside world. *Encapsulation* is the hiding of information.

- ***Definition 3:*** **Data Abstraction** is the separation between the specification of a data object and its implementation.

Basic Definitions of OOP (Cont.)

- ***Definition 4:*** **Object oriented programming** is a method of implementation in which:
  - Objects are the fundamental building blocks.
  - Each object is an instance of some type (or class).
  - Classes are related to each other by inheritance relationships. (Programming methods that do not use inheritance are not considered to be object-oriented).

- ***Definition 5:*** A language is said to be **Object oriented language** if:
  - It supports objects.
  - It requires objects to be belong to a class
  - It supports inheritance.

Evolution of Programming Language and History of C++

Higher order programming languages may be classified into four generation:

- **First Generation Languages (1GL):** An example of this is FORTRAN, these languages have the *ability to evaluate mathematical expressions.*
- **Second Generation Languages (2GL):** Examples of these include Pascal and C. The emphasis of these languages is on *efficiently expressing algorithms.*
- **Third Generation Languages (3GL):** Examples of these include Modula and Ada. These languages *introduced the concept of abstract data types.*
- **Fourth Generation Languages (4GL)--Object-Oriented languages:** Examples of these include Smalltalk, Objective C and C++. These *languages emphasis the expression of the relationship between abstract data types through the use of inheritance.*

Fundamental Concepts of OOP:

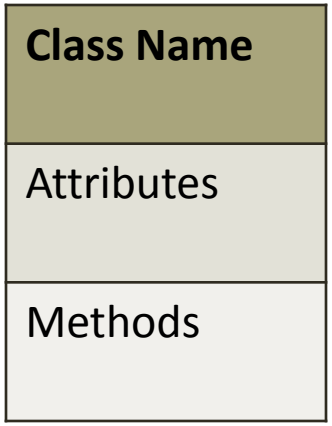
1) Objects
2) Classes
3) Encapsulation
4) Inheritance
5) Polymorphism

Classes and objects

- *A class: is a general description of an object.*
- *A class: is a set of objects that share common conceptual basis that includes a set of attributes and a set of operations on these attributes.*
- **Class components:**
  - **Attributes: int – string – float … etc variables**
  - **Methods:**
    - **Constructor &destructor**
    - **Modifier methods (Set functions)**
    - **Access methods (Get functions)**
    - **Other functions (like display , ….)**

What is UML?

- *Unified Modelling Languages (UML):* is a standard notation for the modelling of real-world objects as a first step in developing an object oriented program. It describes one consistent language for specifying, visualizing, constructing, and documenting the artifacts of software systems



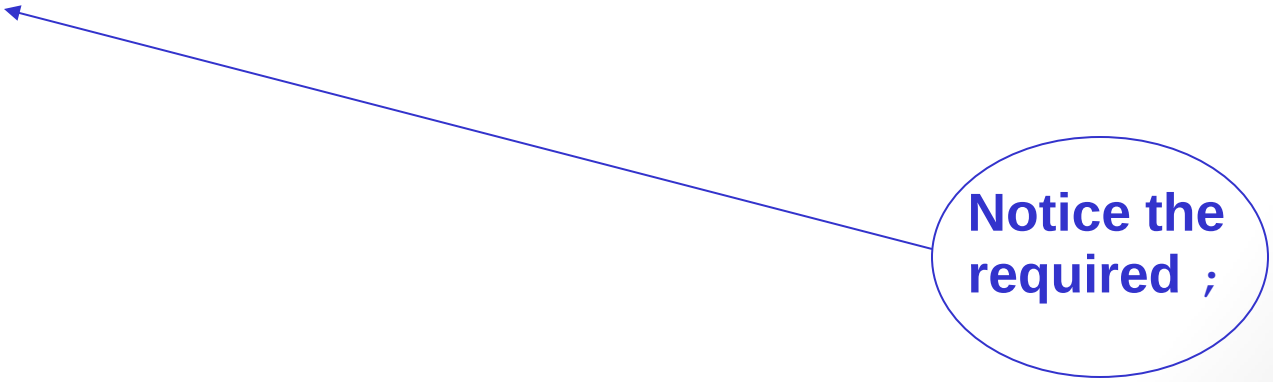
Public Interface

- Class objects are accessed from outside the class via a public interface
  - Some of the member functions can be called from outside the class
  - These member functions provide access to the class object's data from outside of the class
  - This provides some protection from data corruption

Introduction to Classes

- A class declaration describes the member variables and member functions that its objects will have
- It is a pattern for creating objects
- Class Declaration Format:

```
class className
{
    declaration;
    declaration;
};
```



Access Specifiers

- Used to control access to members of the class.
- Each member is declared to be either

`public`: can be accessed by functions outside of the class

or

`private`: can only be called by or accessed by functions that are members of the class

or

`protected`: used with inheritance

More on Access Specifiers

- Can be listed in any order in a class
- Can appear multiple times in a class
- If not specified, the default is **`private`**

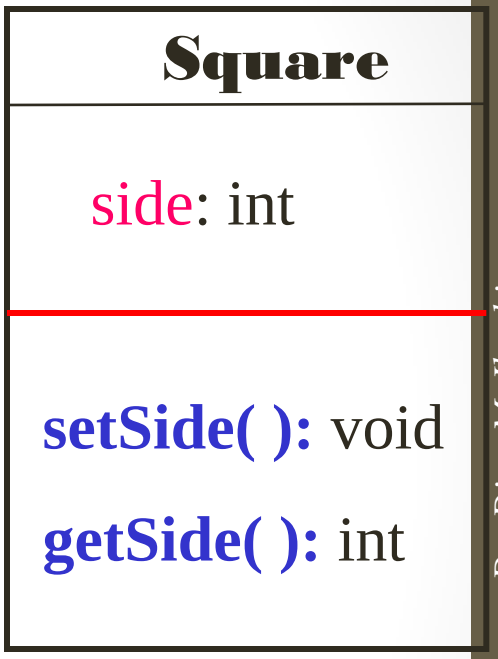
Class Example

```
class Square
{
    private:
      int side;

    public:
      void setSide(int s)
      { side = s; }

      int getSide()
      { return side; }
};
```

Access specifiers



Introduction to Objects

- An object is an instance of a class
- Defined like structure variables

```
Square sq1, sq2;
```

- Access members using dot operator

```
sq1.setSide(5);
cout << sq2.getSide();
```

- Compiler error if attempt to access **`private`** member using dot operator

Defining Member Functions

- Member functions are part of a class declaration
- Can place entire function definition inside the class declaration

or

- Can place just the prototype inside the class declaration and write the function definition after the class

Defining Member Functions Inside the Class Declaration

- Member functions defined inside the class declaration are called inline functions
- Only very short functions, like the one below, should be inline functions

```
int getSide()
{ return side; }
```

Inline Member Function Example

```
class Square
{
   private:
      int side;
   public:
      void setSide(int s)
      { side = s; }
      int getSide()
      { return side; }
};
```

Thanks